جامعة طرابلس ـ كلية تقنية المعلومات

# *Design and Analysis Algorithms*

# تصميم و تحليل خوارزميات

## ITGS301

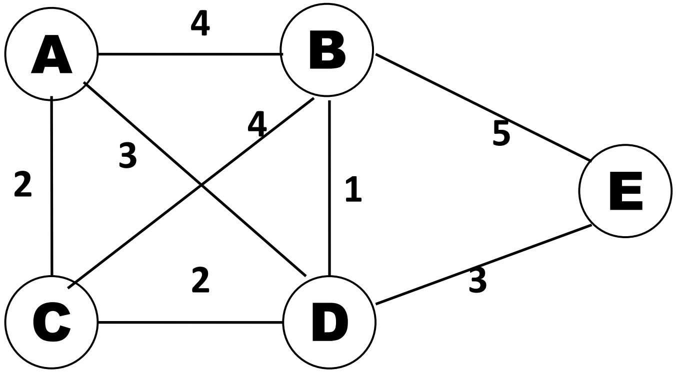
Lecture 11 : المحاضرة الحادية عشر

Tripoli University - Faculty of Information Technology
Marwa Solla
m.solla@uot.edu.ly

# Greedy Method

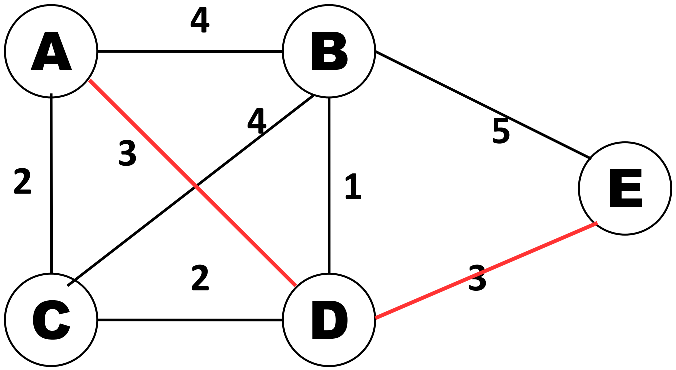
- Chooses the ***Locally Optimal Choice*** hoping that it would give a Globally Optimal Solution.
- Powerful & works for many problems.
- Examples: Minimum spanning trees: **Kruskal's Algorithm and Primm's Algorithm**

**Example:** Find the shortest path from A to E using the Greedy Method



The optimal solution is A → D → E

Path cost is 3+3 = 6



# Graph Algorithms :Kruskal's Algorithm

Help us to find the minimal spanning tree (T).

**Basic idea**

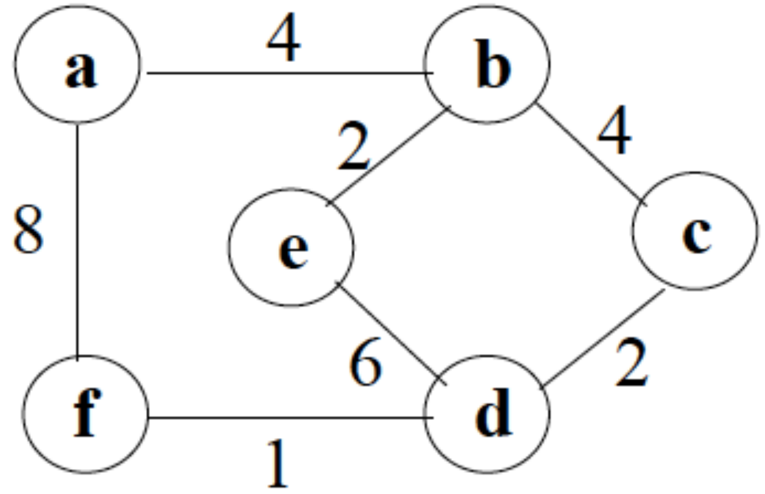
step 1: Arrange all edges in a list (L) in ascending order of weights.
step 2: select the edge of least weight to be part of set T, avoid cycle.
step 3: Repeat step 2 until T becomes a tree that covers all vertices

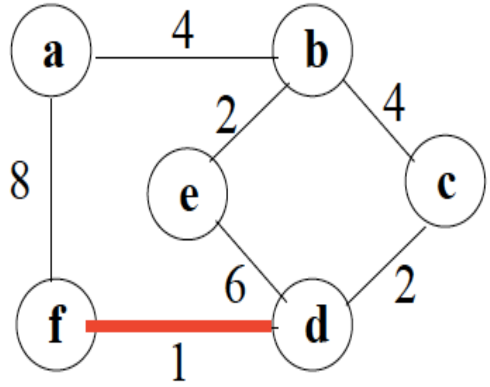
# Algorithm

G: Graph
W: weight
T: Tree

```
MST-Kruskal(G,w)
    T ← Ø
    for each vertex v ∈ V[G]
        Make-Set(v)  // Make separate sets for vertices
    sort the edges by increasing weight w
    for each edge (u,v) ∈ E, in sorted order
        if Find-Set(u) ≠ Find-Set(v) // If no cycles are formed
            T ← T U {(u,v)} // Add edge to Tree
            Union(u,v) // Combine Sets
return T
```

Example 1: Find the minimum spanning tree (MST) from the following graph using Kruskal's Algorithm.






## Initially

**T** = Φ

**Sets** – {a} {b} {c} {d} {e} {f}

**E** (Sorted in Ascending Order )

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |

## Step 1

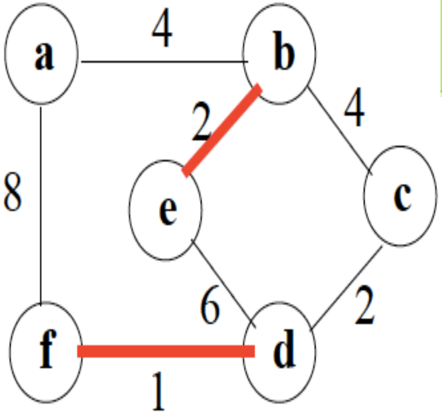
**Take (f,d);** Set(f) ≠ Set(d) => **Add** (f,d) to T, **Combine** Set(f) & Set(d)

**T** = {(f,d)}

**Sets** – {a} {b} {c} {e} {f,d}

## Step 2

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |



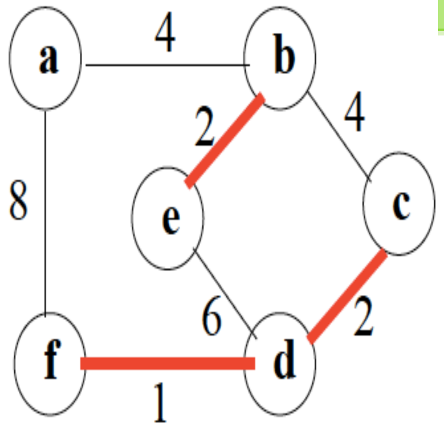

**Take (b,e);** Set(b) ≠ Set(e) => **Add** (b,e) to T, **Combine** Set(b) & Set(e)

T = {(f,d), (b,e)}

**Sets –** {a} {b,e} {c} {f,d}

## Step 3

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |



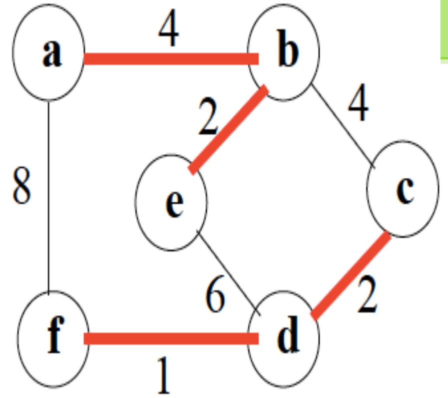

**Take (c,d);** Set(c) ≠ Set(d) => **Add** (c,d) to T, **Combine** Set(c) & Set(d)

T = {(f,d), (b,e), (c,d)}

**Sets –** {a} {b,e} {f,d,c}

## Step 4

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |



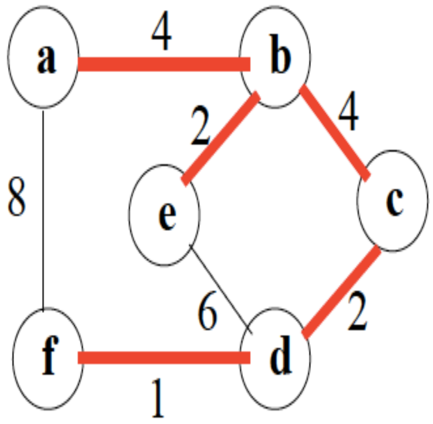

**Take (a,b);** Set(a) ≠ Set(b) => **Add** (a,b) to T, **Combine** Set(a) & Set(b)

T = {(f,d), (b,e), (c,d), (a,b)}

**Sets –** {b,e,a} {f,d,c}

## Step 5

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |



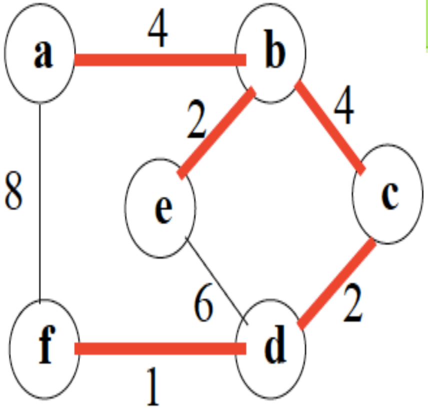

**Take (b,c);** Set(b) ≠ Set(c) => **Add** (b,c) to T, **Combine** Set(b) & Set(c)

**T** = {(f,d), (b,e), (c,d), (a,b), (b,c)}

**Sets –** {b,e,a,f,d,c}

## Step 6

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |



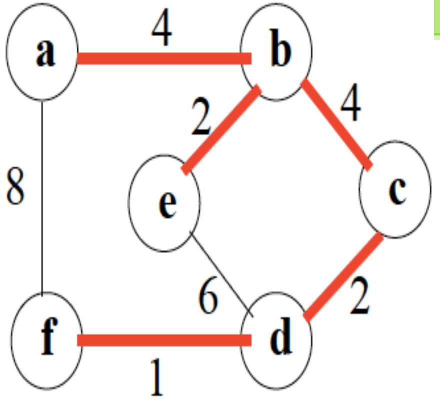

**Take (e,d);** Set(e) = Set(d) => **Ignore**

T = {(f,d), (b,e), (c,d), (a,b), (b,c)}

**Sets –** {b,e,a,f,d,c}

## Step 7

| | | | | | | |
|---|---|---|---|---|---|---|
| (f,d) | (b,e) | (c,d) | (a,b) | (b,c) | (e,d) | (a,f) |



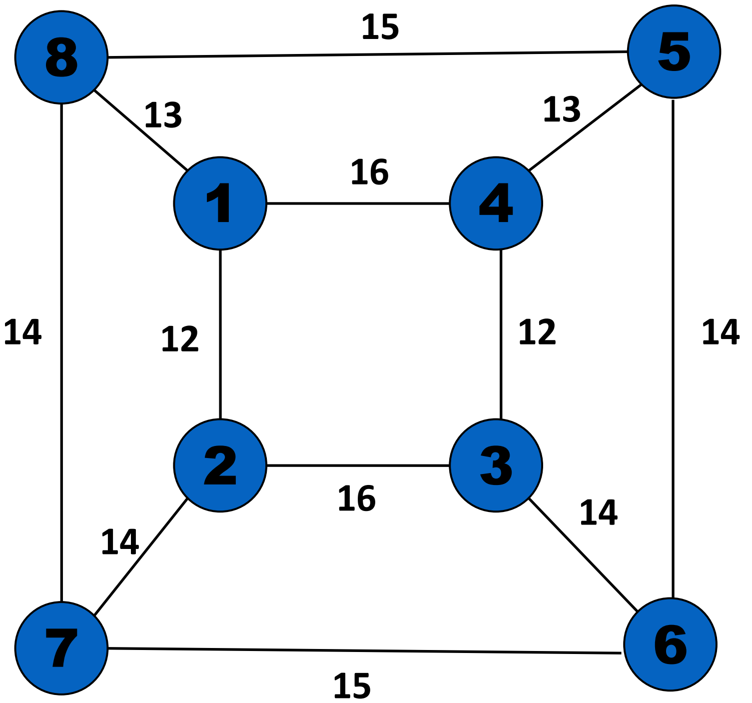

**Take (a,f);** Set(a) = Set(f) => **Ignore**

**T** = {(f,d), (b,e), (c,d), (a,b), (b,c)}

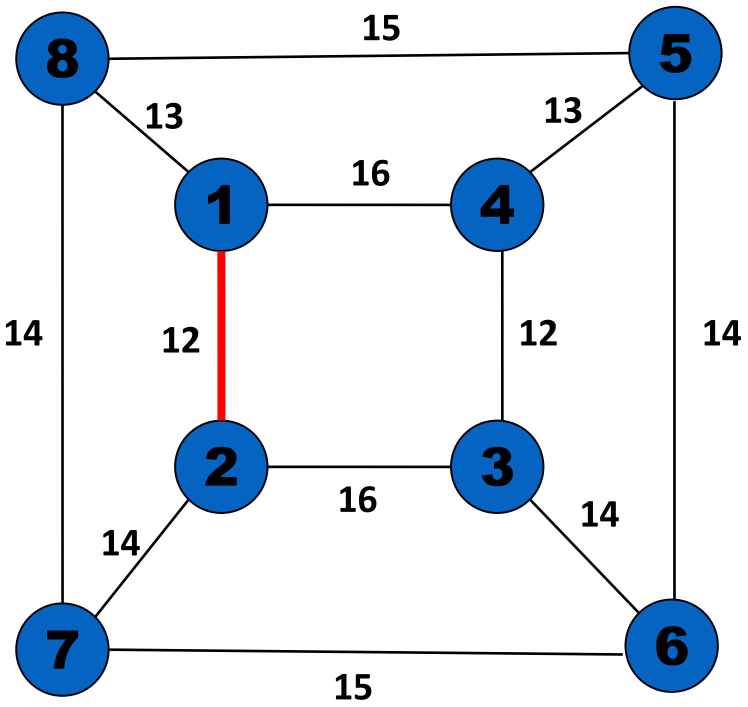
**Sets –** {b,e,a,f,d,c}

MST = 1+2+2+4+4 = 13

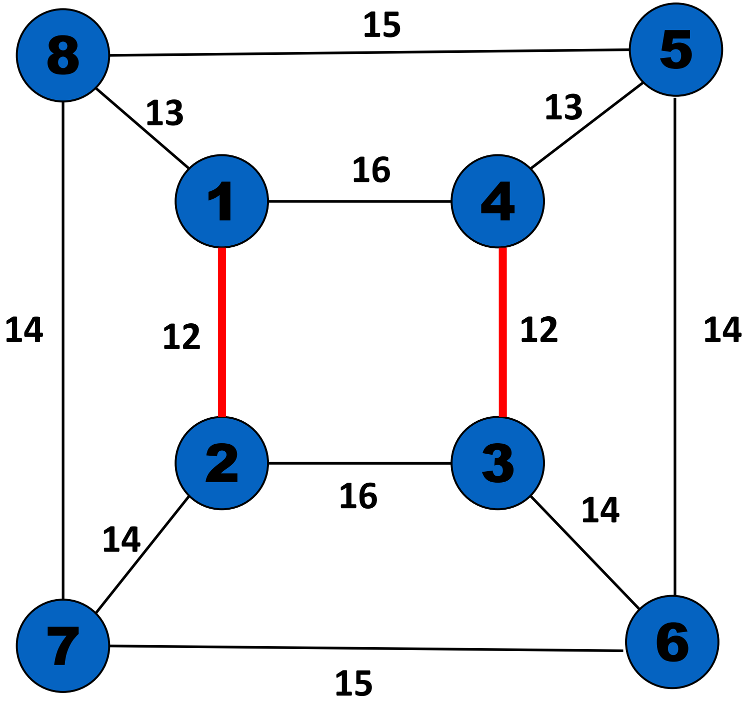
## Example 2: Find the minimum spanning tree (MST) from the following graph using Kruskal’s Algorithm.



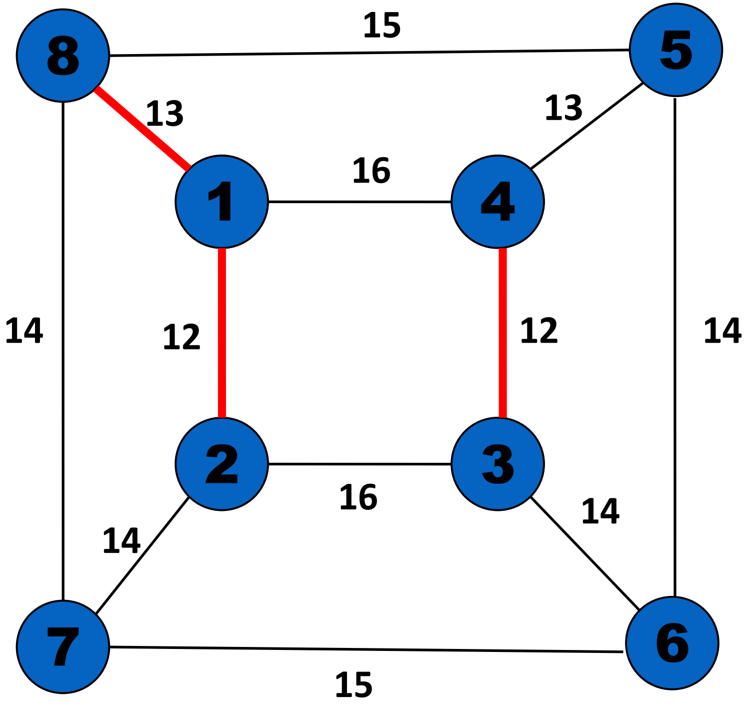

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |



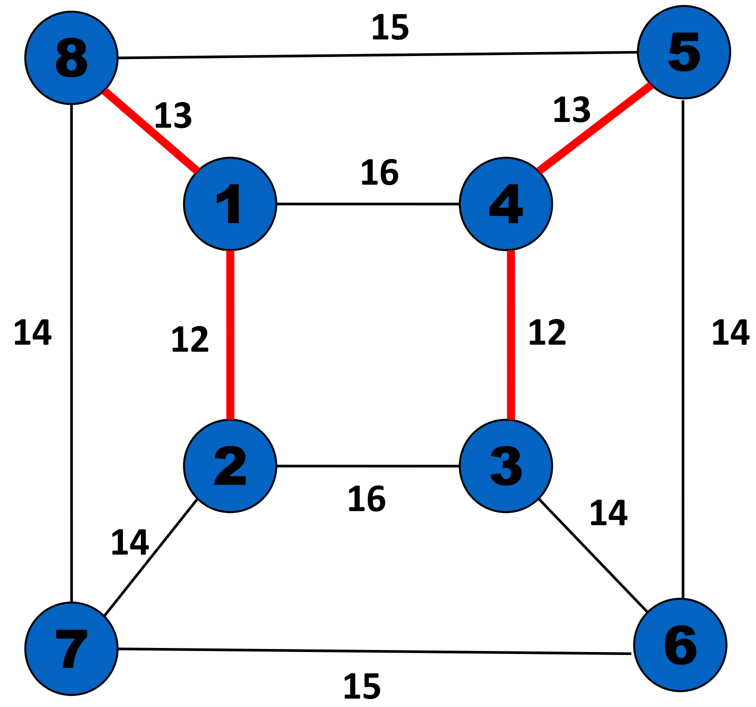

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |



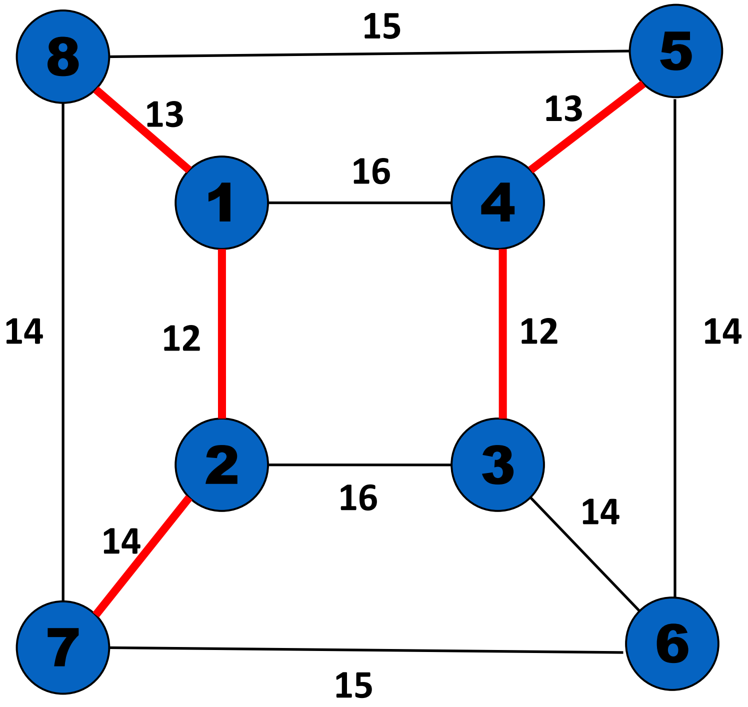

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |



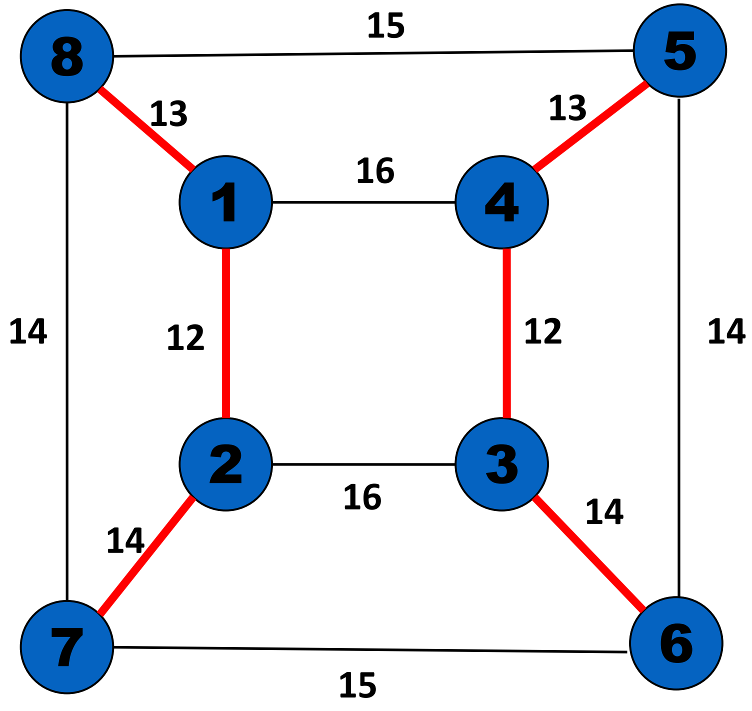

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |



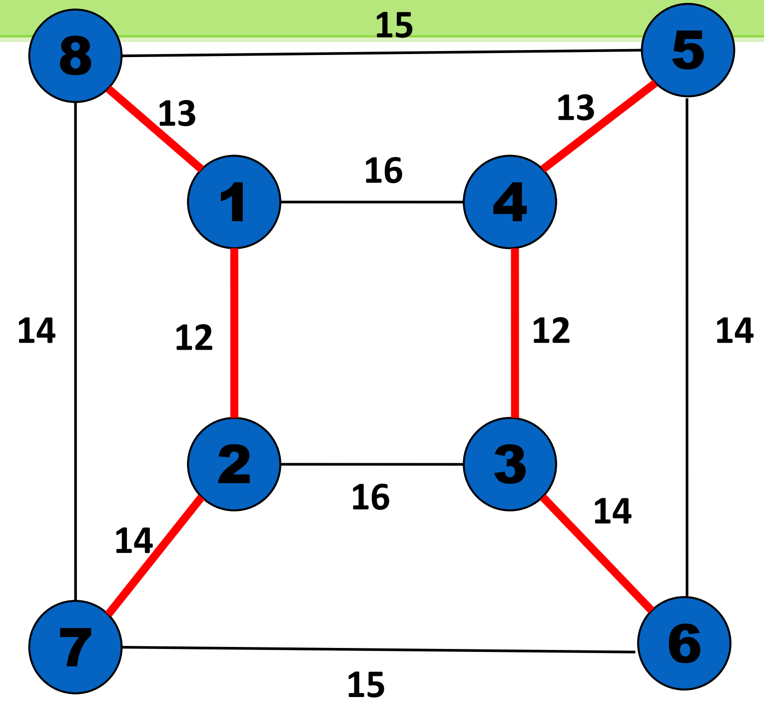

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |




| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |



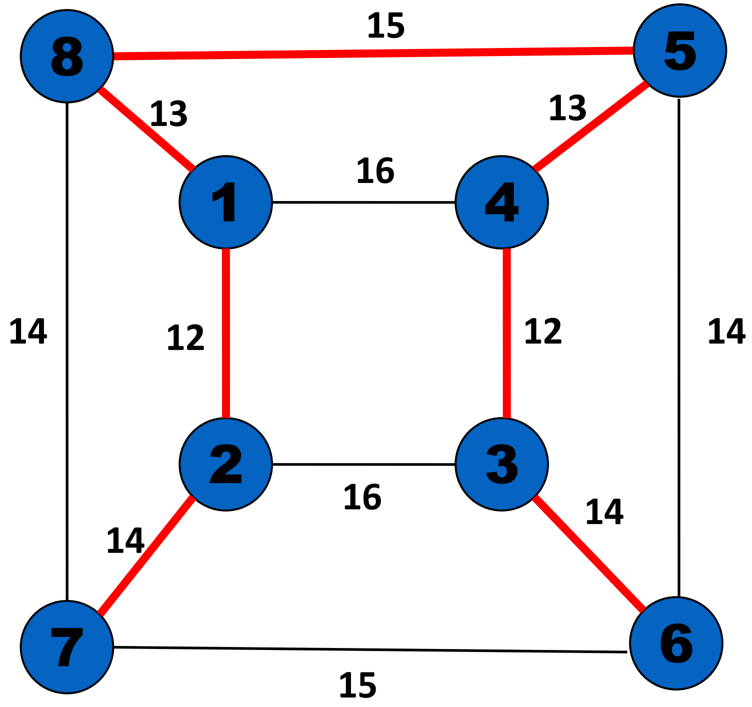

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |



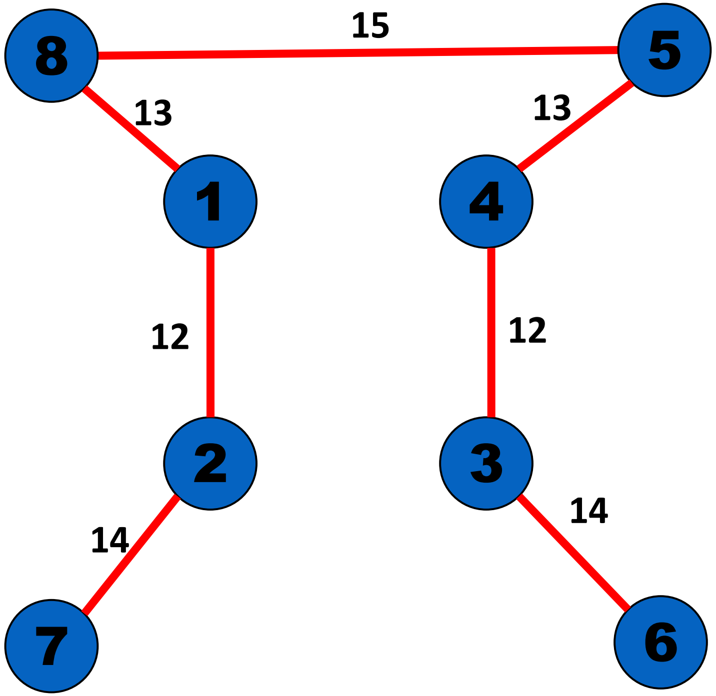

| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |

skip
skip

**Skip {7,8} and {5,6} to avoid cycle**




| | |
|---|---|
| {1,2} | 12 |
| {3,4} | 12 |
| {1,8} | 13 |
| {4,5} | 13 |
| {2,7} | 14 |
| {3,6} | 14 |
| {7,8} | 14 |
| {5,6} | 14 |
| {5,8} | 15 |
| {6,7} | 15 |
| {1,4} | 16 |
| {2,3} | 16 |




MST = 12+12+13+13+14+14+15 = 93

# Graph Algorithms :Prim's Algorithm

Used to find the minimal spanning tree (T) in connected graph.

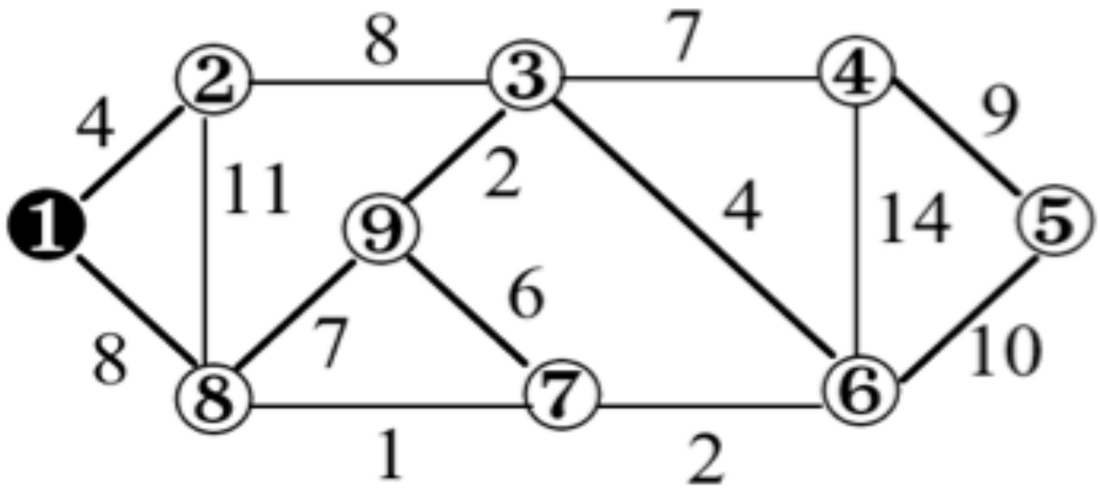
## Basic idea

step 1: choose any starting vertex, look at all edges connected to the vertex and choose the one with lowest weight and add it to the Tree (T).

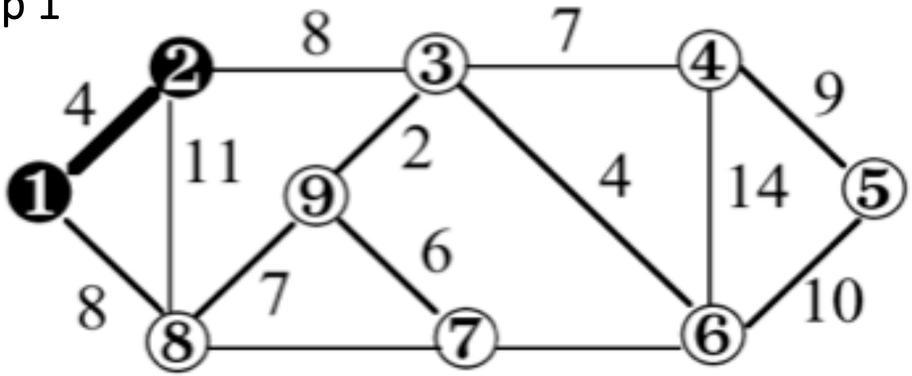
step 2: look at all edge connected to the tree, **choose the one with** of least weight to be part of set T, avoid cycle. (if more than one choose at random)

step 3: Repeat step 2 until T becomes a tree that covers all vertices

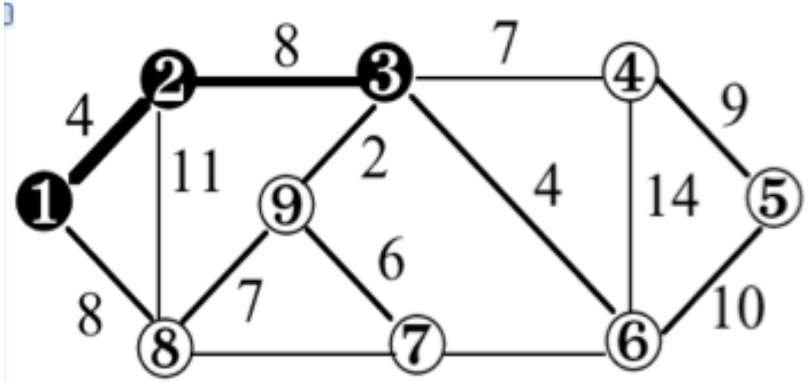
# Example 1:



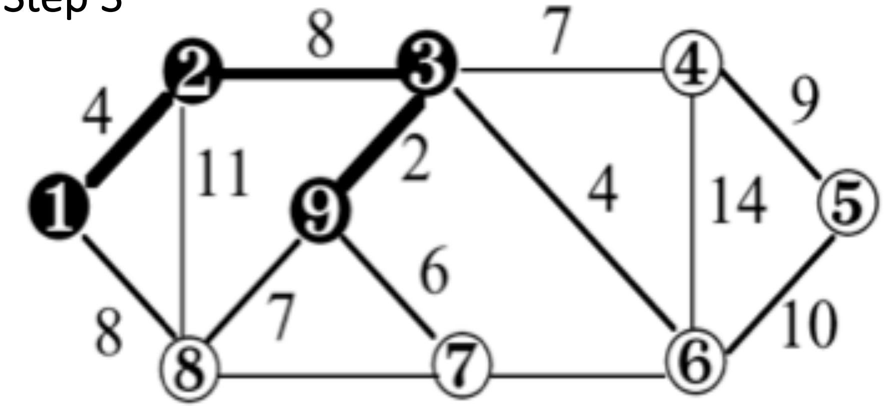
Step 1



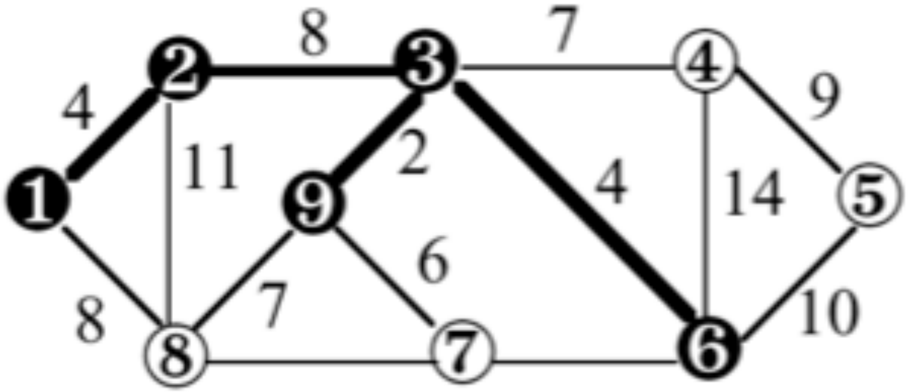

Step 2



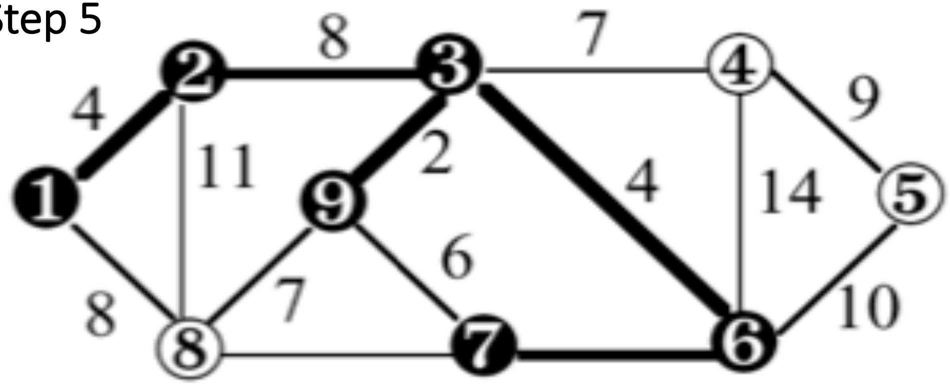
Step 3



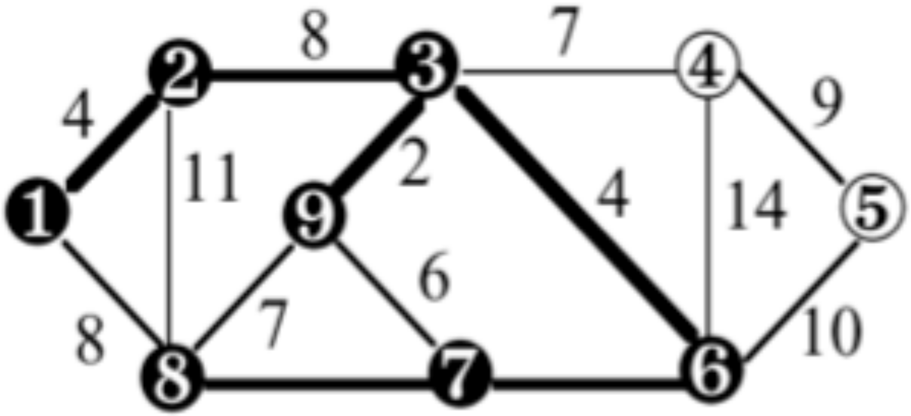

Step 4



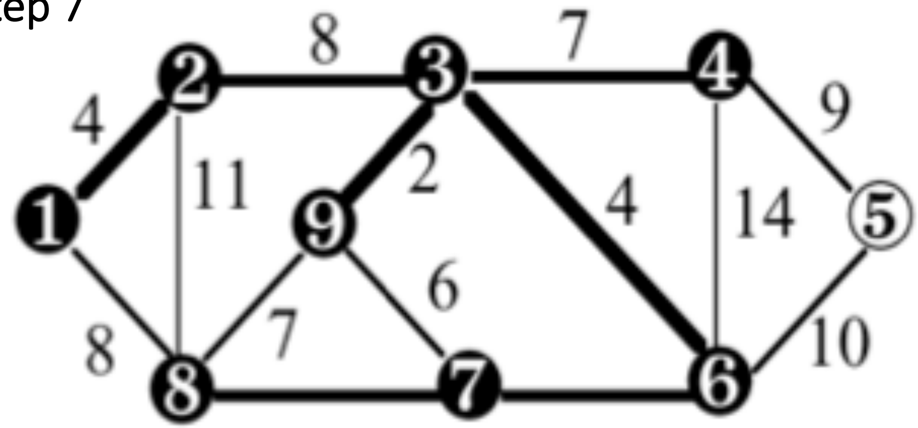
Step 5



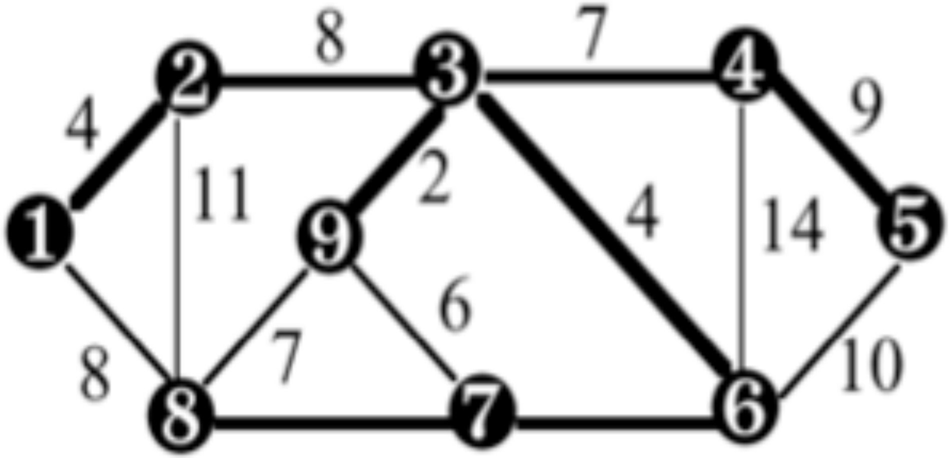

Step 6

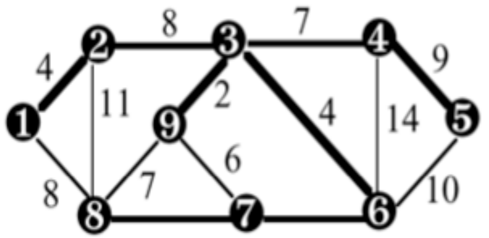


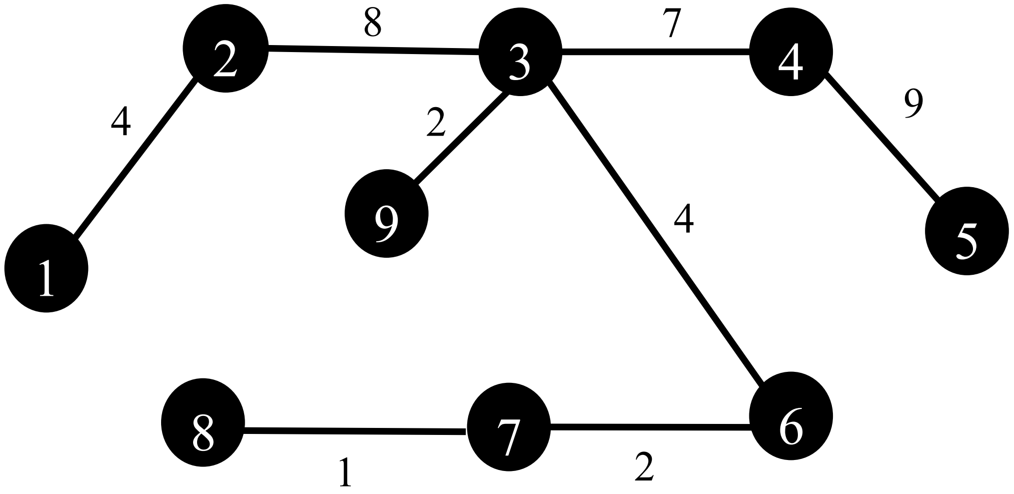
Step 7




Step 8

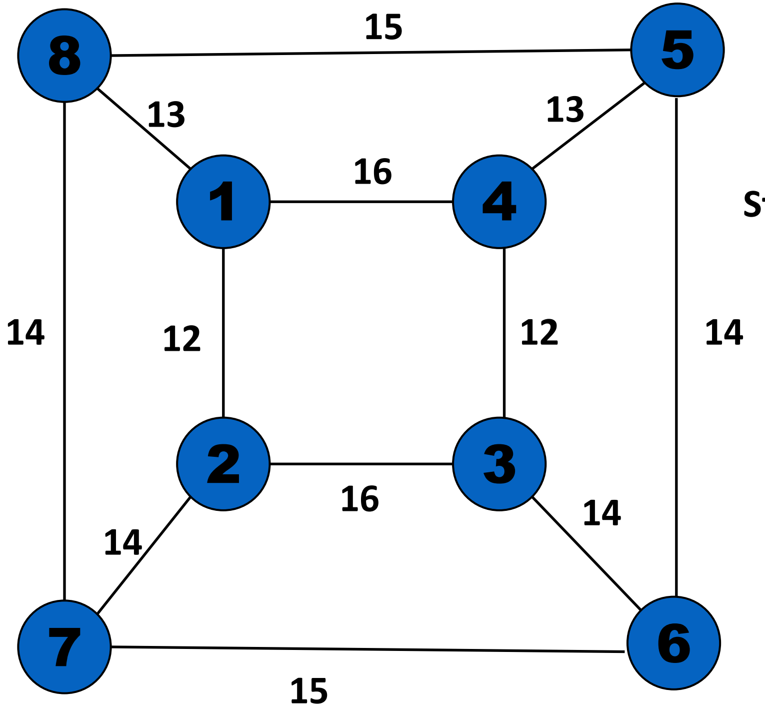




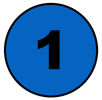



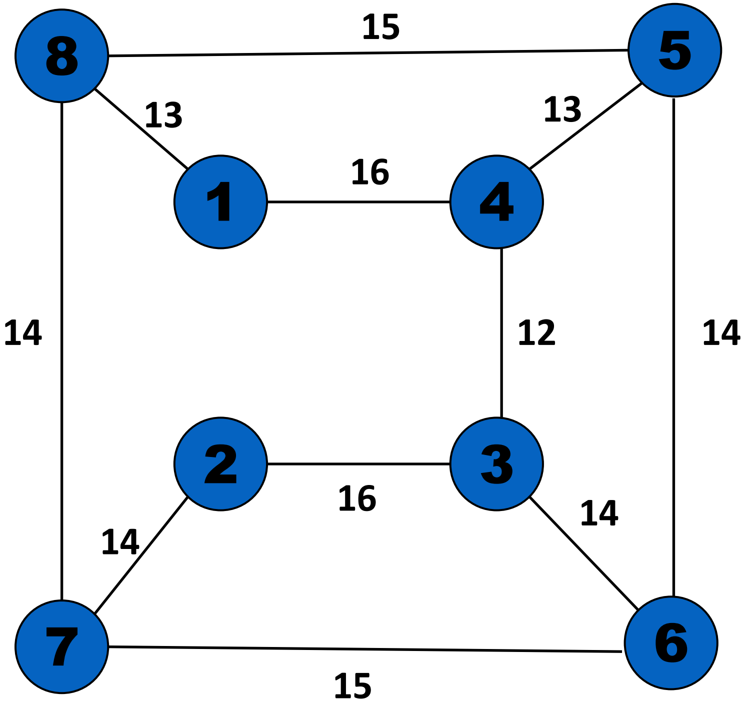

**MST = 37**

Example 2: Find the minimum spanning tree (MST) from the following graph using Prim's Algorithm.

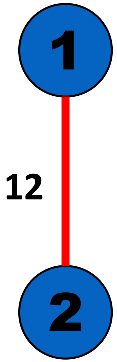


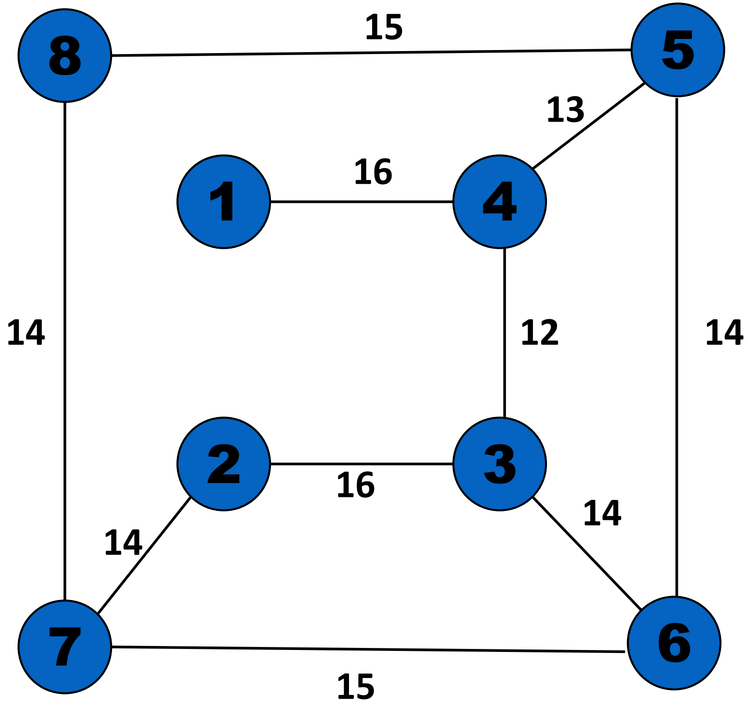

**Start from any arbitrary vertex**



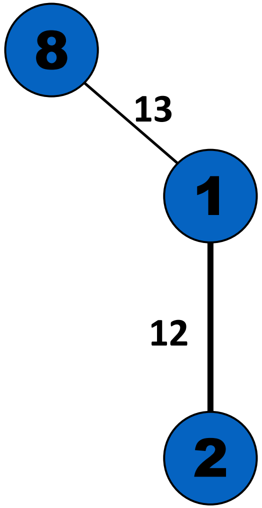



The best choice is {1,2}

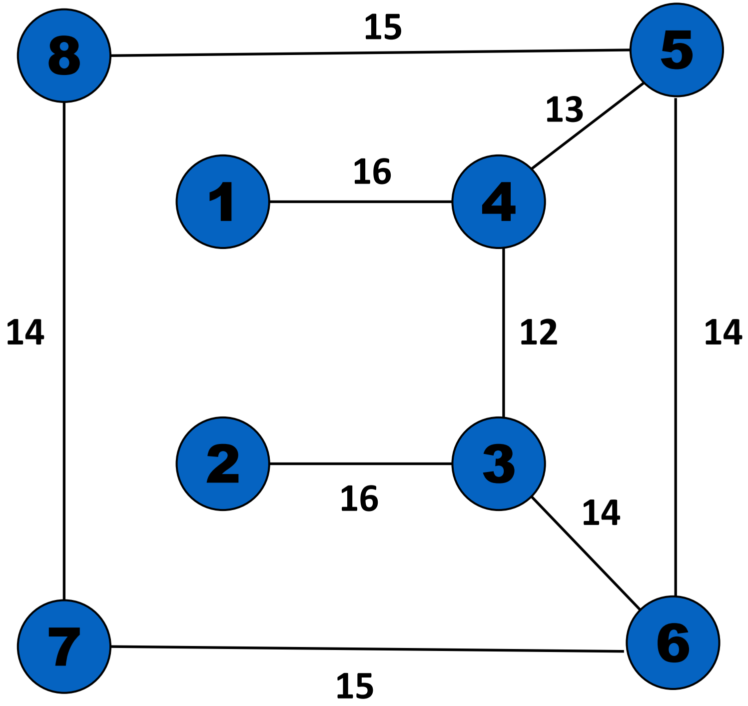


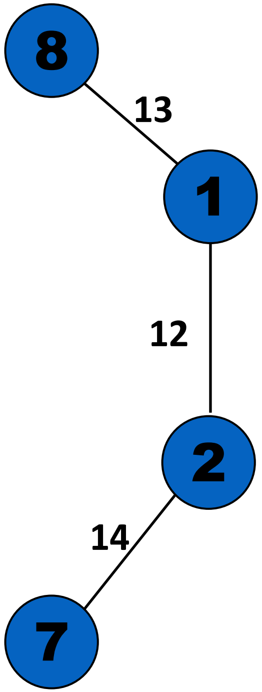



After {1,8} we may choose {2,7} or {7,8}

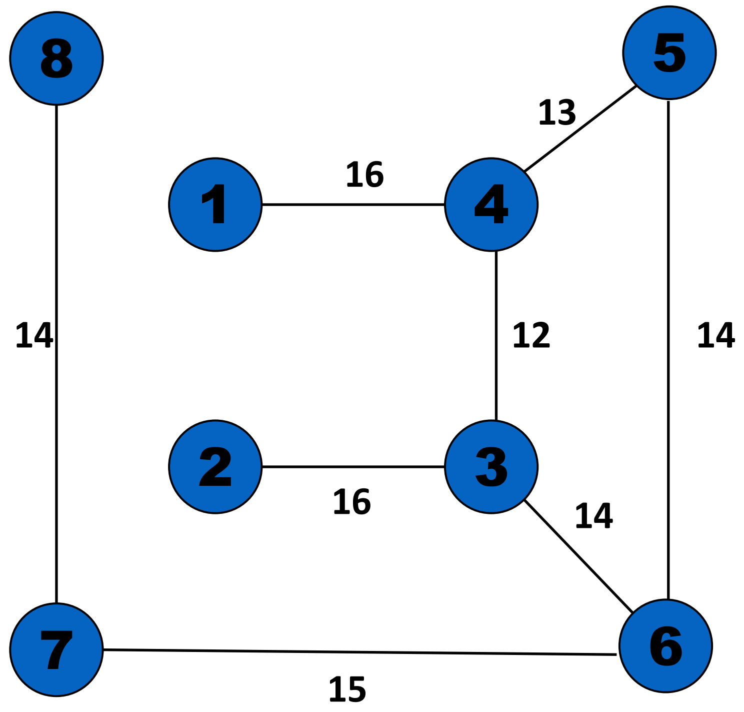


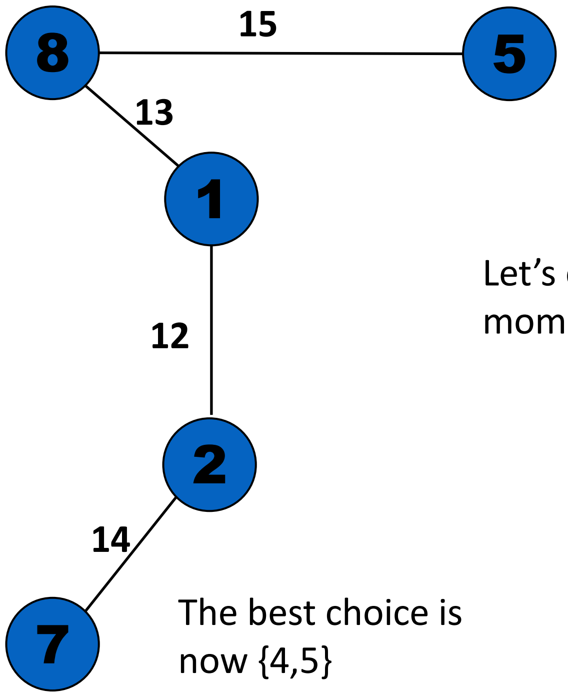

There are more than one MST



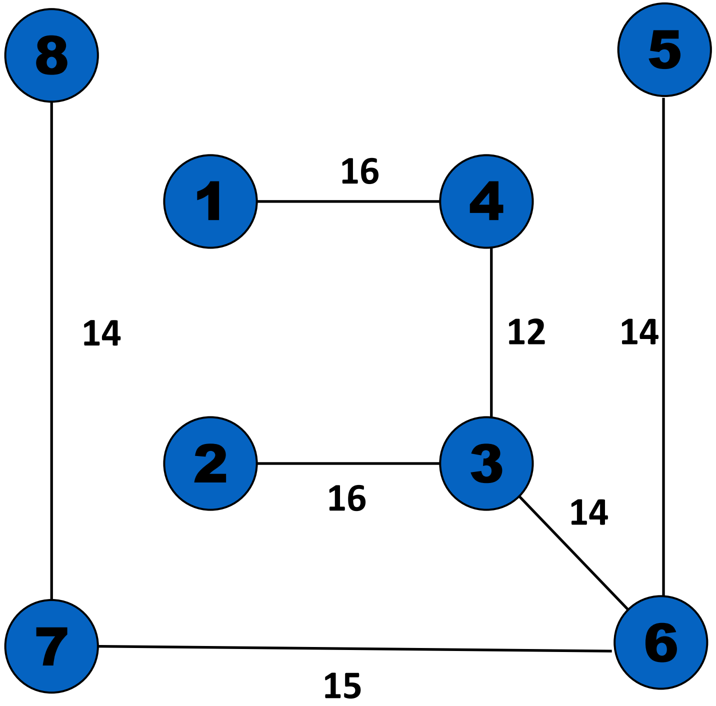

Let’s choose {2, 7} at this moment .
Prim’s algorithm 12 We are free to choose {5,8} or {6,7} but not {7,8} because we need to avoid cycle

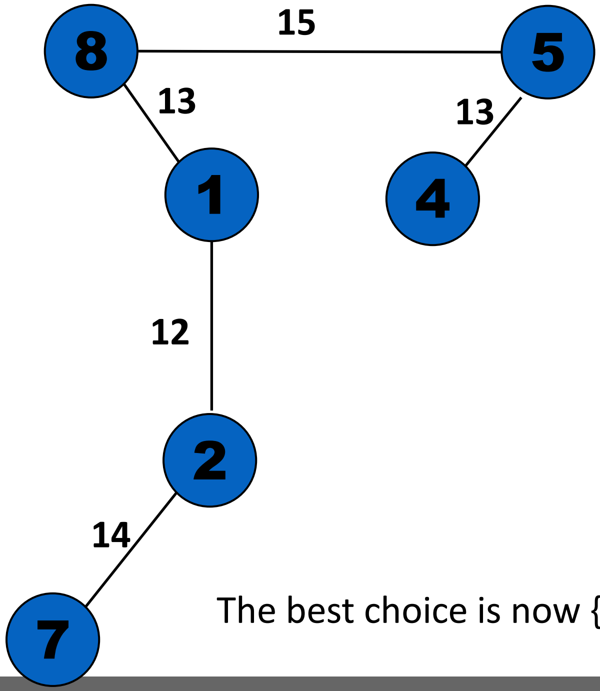


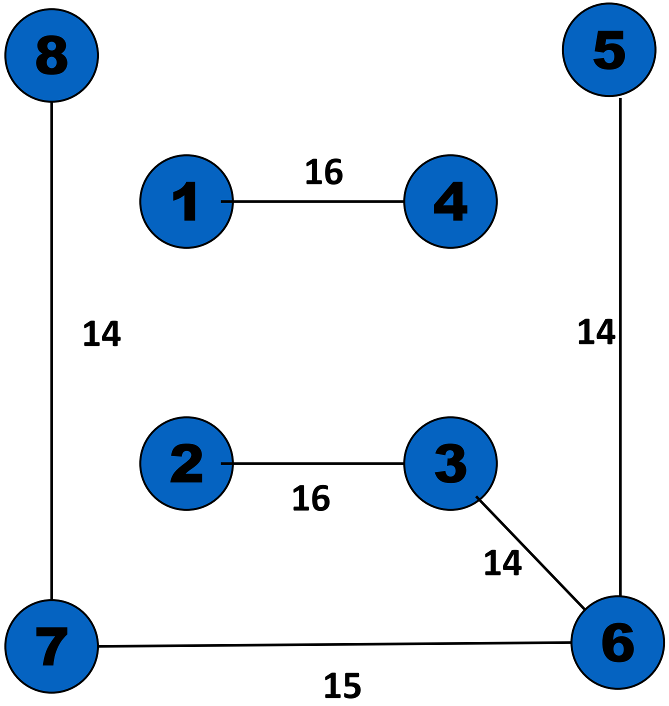

Let's choose {5,8} at this moment

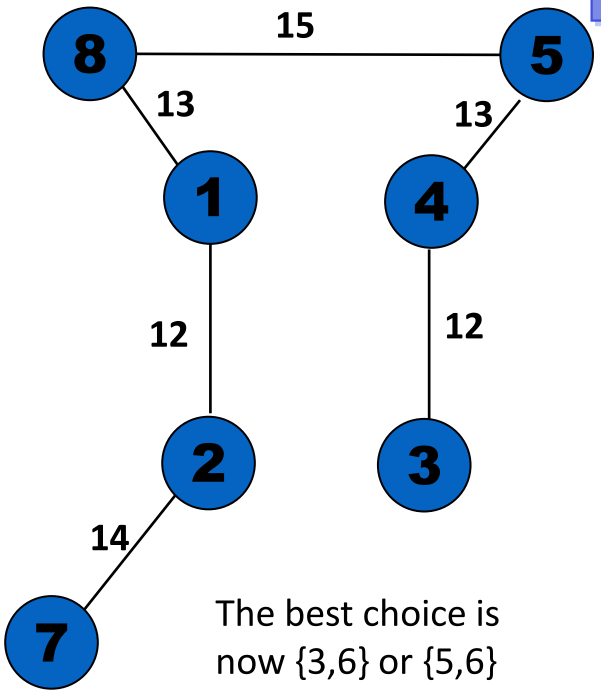
The best choice is now {4,5}



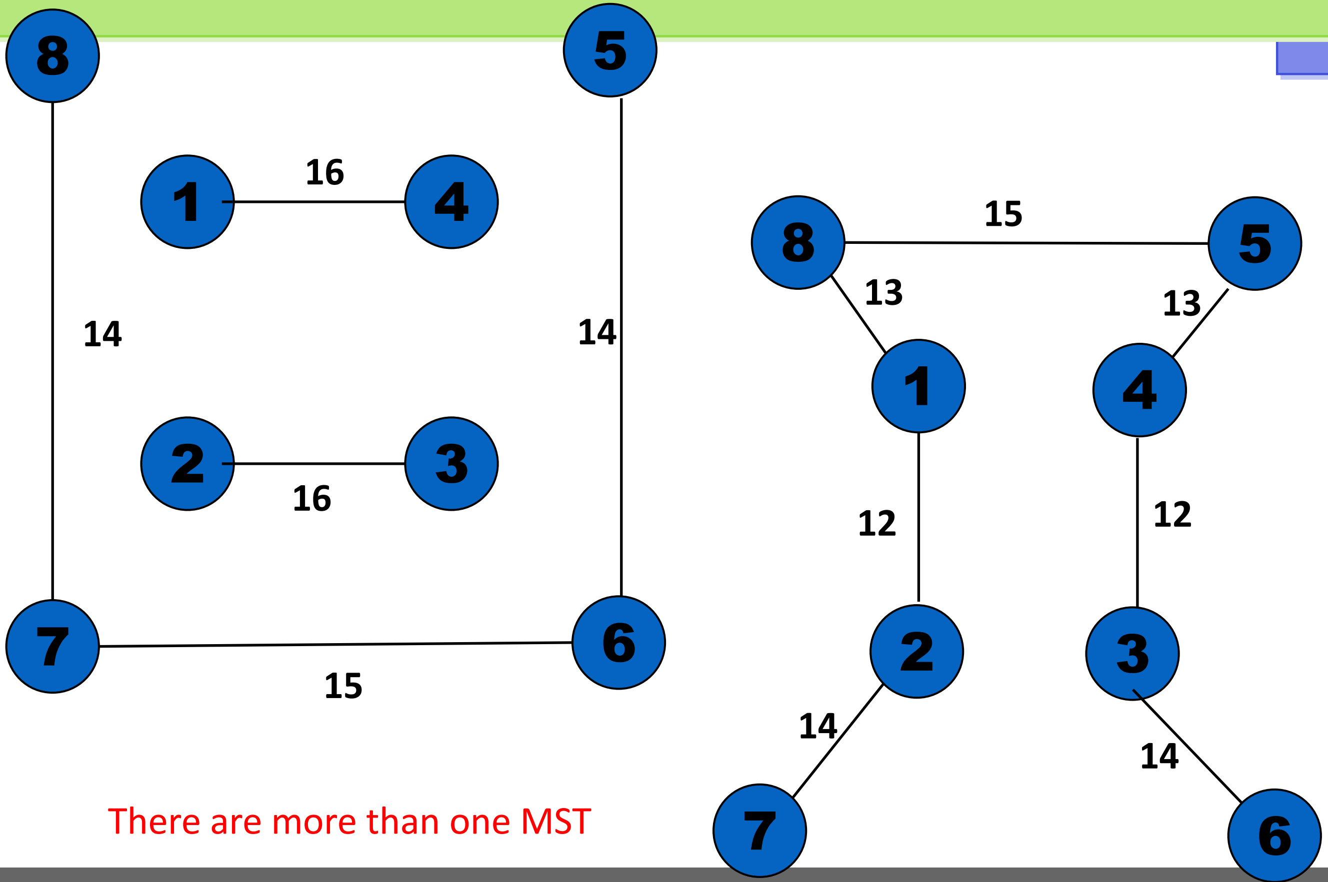

The best choice is now {4,3}



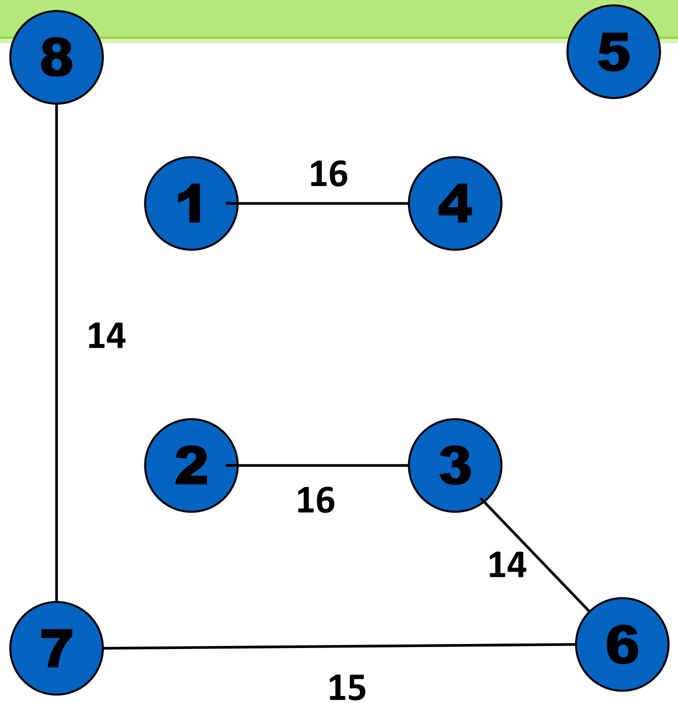



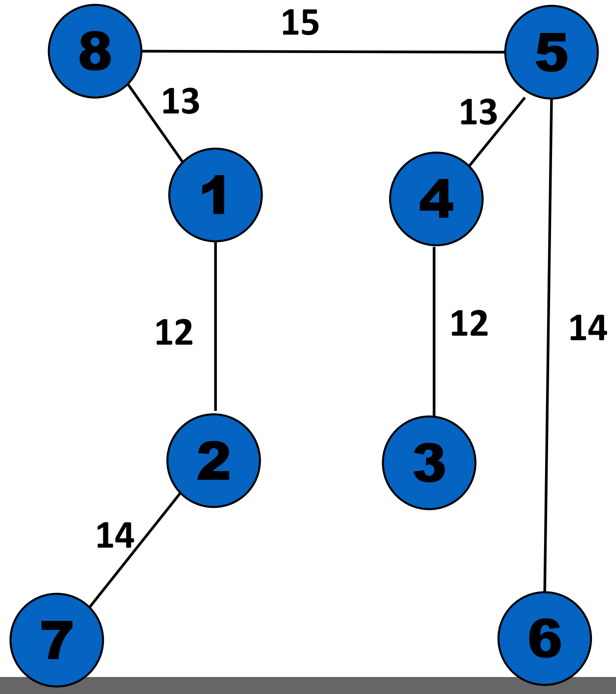

The best choice is now {3,6} or {5,6}

There are more than one MST

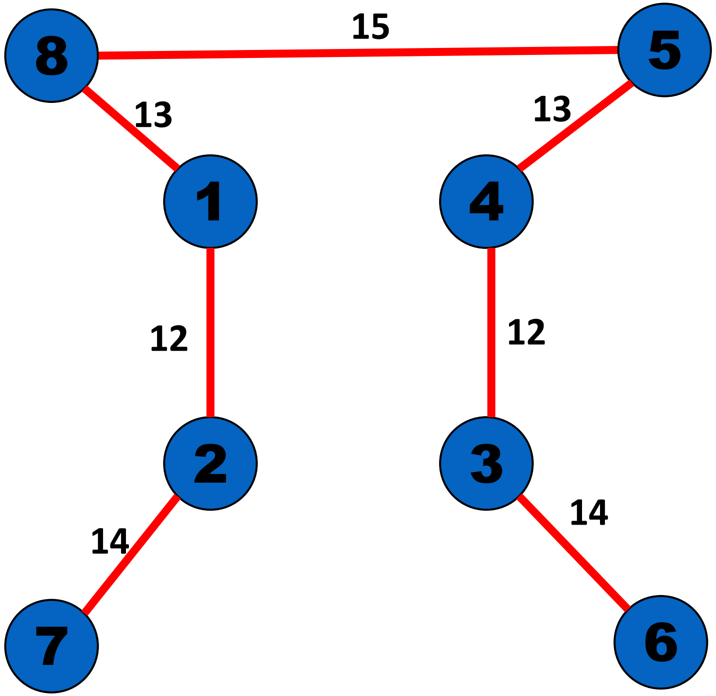



There are more than one MST




There are more than one MST




MST = 12+12+13+13+14+14+15 = 93

***The End .***